# Remembrance Agent: A continuously running automated information retrieval system

**Bradley J. Rhodes**
MIT Media Lab, E15-305
20 Ames St.
Cambridge, MA 02139 rhodes@media.mit.edu

**Thad Starner**
MIT Media Lab, E15-394
20 Ames St.
Cambridge, MA 02139 thad@media.mit.edu

**Abstract**
The Remembrance Agent (RA) is a program which augments human memory by displaying a list of documents which might be relevant to the user's current context. Unlike most information retrieval systems, the RA runs continuously without user intervention. Its unobtrusive interface allows a user to pursue or ignore the RA's suggestions as desired.

## The Remembrance Agent

Most of a desktop computer's time is spent waiting: waiting for the user to hit the next keystroke, waiting for the user to read the next page, and waiting for the next packet to come down the network. The remembrance agent uses those wasted CPU cycles constructively by performing continuous searches for information that might be of use in its user's current situation. For example, while an engineer reads email about a project the remembrance agent reminds her of project schedules, status reports, and other resources related to the project in question. When she stops reading email and starts editing a file, the RA automatically changes it recommendations accordingly. These suggestions are presented in the form of one-line summaries at the bottom of the screen. Here they can be easily monitored, but won't distract from the primary work at hand. The full text of a suggestion can be brought up with a single keystroke.

Most applications for augmenting human memory, e.g. those developed by (Jones 1986) and (Lamming & Flynn 1994), have concentrated on recall-on-demand. For example, they can answer questions such as "when is that conference paper deadline?". However, they do not address situations where a user does not remember enough to realize he or she has forgotten something in the first place. This associative form of recall is what reminds a user that an important conference exists at all. The Remembrance Agent augments this continuous, associative form of recall.

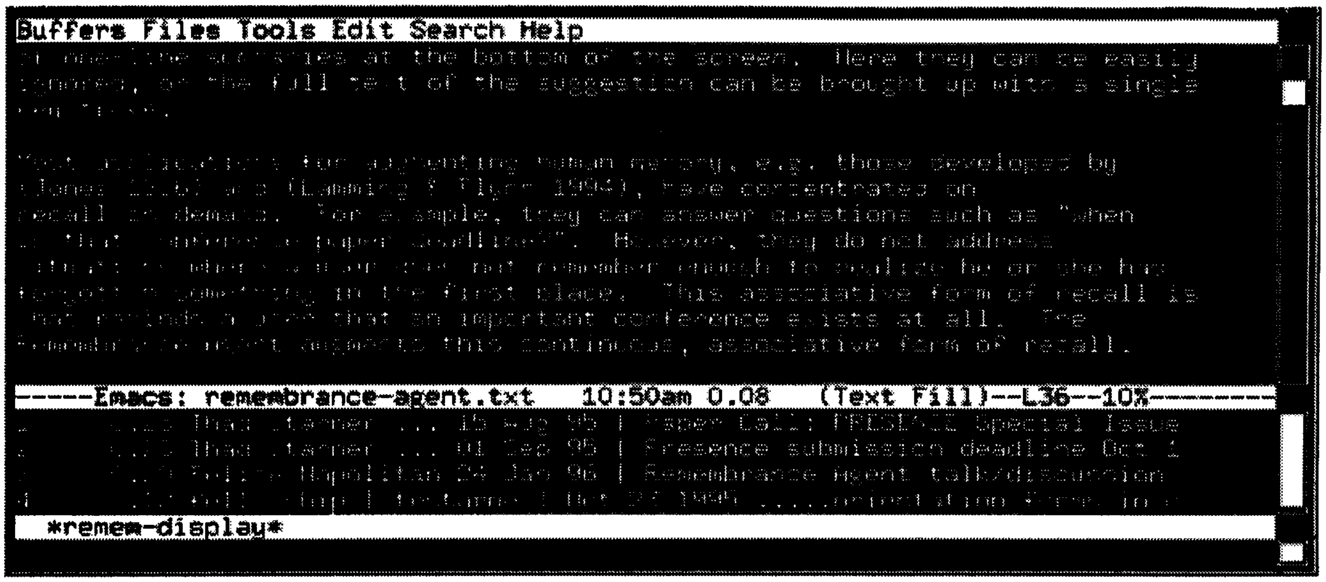
Beyond simple memory enhancement, the RA also suggests alternative ways to organize knowledge. The connection between a suggested document and the user's current context may not be obvious to the user until pointed out. In this way the RA both helps the user organize their own datafiles into new categories, and helps create continuous brainstorming session where new ideas and possible connections are suggested.

## Implementation

The RA front-end runs in elisp under Emacs-19, a UNIX based text editor which can also be used for applications such as email, netnews, and web access. The front end displays one-line suggestions along with a numeric rating indicating how relevant it thinks the document is. It also brings up the full text of suggested documents when requested. Every few seconds it collects the text within certain ranges around the current cursor position and sends this text to the information retrieval program.

The back-end is a program which, given a query-text, produces suggestions of similar documents from a pool of documents which are pre-indexed nightly. The current implementation uses the SMART information retrieval program, which decides document similarity based on the frequency of words common to the query and reference documents (Salton & Lesk 1971). While SMART is not as sophisticated as many more modern information-retrieval systems, it has the advantage that it requires no human preprocessing of the documents being indexed.

In the current version the RA utilizes two sources of suggestions. The first three lines print suggestions from the last year's worth of personal email (a little over 60 Megabytes worth). The last line prints suggestions from about 7 Megabytes of notes files entered over the past few years. The



```
Buffers Files Tools Edit Search Help
[illegible] the [illegible] at the bottom of the screen.  Here they can be easily
ignored, or the full text of the suggestion can be brought up with a single
[illegible].

[illegible] for augmenting human memory, e.g. those developed by
(Jones [illegible]) and (Lamming & Flynn 1994), have concentrated on
recall on demand.  For example, they can answer questions such as "when
[illegible] paper deadline?".  However, they do not address
[illegible] not remember enough to realize he or she has
forgotten something in the first place.  This associative form of recall is
[illegible] that an important conference exists at all.  The
[illegible] supports this continuous, associative form of recall.

-----Emacs: remembrance-agent.txt   10:50am 0.08   (Text Fill)--L36--10%--------
1      [illegible] Thad Starner ... 15 Aug 95 | Paper Call: PRESENCE Special Issue
2      [illegible] Thad Starner ... 01 Sep 95 | Presence submission deadline Oct 1
3      [illegible] Napolitan 24 Jan 96 | Remembrance Agent talk/discussion
4      [illegible]
  *remem-display*
```

**Figure 1: A screen-shot from the Remembrance Agent**

short, one-line description of email messages contains who the message was from, when it was sent, and the subject line of the message. The one-line description of notes files contains the filename, the owner of the file, date last modified, and the first few words of the document.

The RA should be able to suggest documents similar to tangents in a document as well as those similar to the whole. At different times a user might be interested in suggestions relating to the person who's name was just typed, to the last paragraph typed, or relating to a much broader range. To achieve this, three "scopes" are defined for a document, centered around the current cursor location. One scope is dedicated to those suggestions relevant to the last thousand words (usually the entirety of the document). Another is dedicated to the last fifty words, and a third to the last ten words. Each scope accesses either the email or text database, and is allotted one or more display lines. The number of words each scope covers, the number of display lines each uses, the time between updates, and the database each points to are all customizable by the user.

## Design Issues

There are three different ways of handling the timing of an automated task. The first is to perform a task only when specifically requested. Spell-checking a file and performing a web-search fall into this category. A second way is for an automated task to always lurk in the background, but to only act when a specific "trigger" occurs. Such programs include calendar programs that automatically tell you when you're going to be late for a meeting, and programs that alert you when you have new email waiting. Finally, there are tasks which are performed continuously, like a clock program or CPU-load meter. The RA falls into this third category.

Unlike programs which warn you about important meetings, it is usually not a major problem when the user misses suggestions presented by the RA. However, since the RA runs continuously, suggestions could quickly distract from the user's primary task if they attracted too much attention. For these reasons the RA's suggestions are kept unobtrusive. Suggestions are kept to a single line each, and are always printed at the bottom of the text-editor window. The full display area is also limited to a maximum of 9 lines, though it defaults to operating with only three or four. Finally, no color cues or highlights were used in the suggestion-display area.

It is often the case that just seeing the one-line description triggers the entire memory, with no need to bring up the full document. Also, in some cases the description line supplies desired information directly (for example, the last name of someone who sent email). For this reason, as much information as possible is packed into the short description line.

While suggestions are unobtrusive, it is easy to access the entirety of a suggested document. To view the document being suggested on a given line, the user simply types "control-c" and the display line-number. It is similarly easy to get back to the primary document.

As an additional feature, the user can explicitly ask for recommendations based on an input string. When this method is used to receive suggestions, all the display lines are used to present the top possible suggestions, and the scope information is ignored. The scope information is similarly ignored when the user reads email. Email messages usually contains a single thought, so suggestions based on the entire email are usually more useful than those based on smaller ranges.

Using the system has shown that suggestions are much more useful when the document being suggested only contains one "nugget" of information, and when that nugget is clearly displayed on its one-line description. This "less is more" approach solves several problems. First, it allows the user to tell what a suggestion contains from its description without having to peruse the entire document. Second, a document with only one primary point is more likely to be a good hit. Documents which address several issues will rarely match the user's situation exactly, but will often partially match. Finally, if a suggested document *is* read, the shorter it is the quicker the user can get on with their primary work. Old email are a good length for RA suggestions.

Several of the applications described in the next section use the RA to suggest documents the user has never seen before. While this may be a powerful application, there are extra pitfalls when using someone else's database. The primary pitfall is that the user will not have the context associated with a given suggestion. This makes it far less likely that the user will recognize a document from its one-line summary, and it will be far more difficult for them to judge whether a suggestion is useful. Another problem is that the user won't necessarily know the original context in which a document was created. For example, they might not know that a set of instructions were made in jest, or were in a subsequent message shown to be incorrect. Similarly, documentation or lectures geared towards one kind of audience may be inappropriate for another, causing the original speaker's intention to be misinterpreted. Finally, there are privacy issues when using someone else's "memories." Even if you access someone's email files with their express permission, the people who sent that person mail might not have approved.

## Future Applications

### Reference advisor for technical papers

One application is to have the RA suggest technical reports on a given subject. When it recommends other researchers' papers, these could be referenced or tagged for later reading. When it recommends one's own old papers, these can be scanned for similar material which might be used in the new paper. Such a system could also recommend conferences where the call-for-papers is similar in content to one's own paper.

### Knowledge transfer

One of the large difficulties facing industry is bringing new members of an existing work-group up to speed quickly. If a work-group created its own knowledge-base, new members could access the group Remembrance Agent. This would not only give the employee access to the group knowledge itself (as would any database), but also to the meta-knowledge of *when* particular information is relevant or valuable. Similar applications would exist wherever "just-in-time training" is required. While this application uses a knowledge base not familiar to the user at first, they will not suffer too greatly from the lack-of-context problem discussed above because the knowledge is focused on their current situation, namely their new position within the group. Any context they do not yet know, they need to learn anyway, and the RA will help them learn it.

### Wearable computing

When running on a desktop computer, an RA can only guess the user's context based on the document they are reading or editing. However, the advent of wearable computing (Starner et al. 1995) will allow RAs to work with much more information and many more situations. Global Positioning Systems (GPS) will let the RA know where the user is, while camera and face recognition will let it know who they're talking to. With this extra information, a (greatly enhanced) RA could know that it is around lunch time, that according to camera input the user is with her lunch date, according to her appointment book she has an appointment with her boss in an hour, and according to her GPS she's downtown. From this information it could recommend several good restaurants known for their fast service that are within a few blocks of downtown, which would be agreeable to her lunch date as well.

### Automatic Hypertexting

In the past year several on-line magazines have appeared, such as HotWired (HotWired), many of which have paper counterparts. One of the values added by the on-line versions of these magazines are hyper-linked text, where a reader can click on a word and get more information on that subject. A future RA could conceivably automatically turn normal email, netnews, or papers into hyper-linked documents, automatically linking hot-words to relevant background information.

### Background checker

Another Web-based RA could perform background checks on people sending mail, referenced in papers, or recognized by a wearable-computer-mounted camera. Such an RA could automatically provide information on a person's employer, job title, phone numbers, and profiles of their interests based on newsgroups they frequent. At the click of a button, the user could access that person's home-page for even more information.

## References

HotWired. http://www.hotwired.com/

Jones, W. 1986. On the Applied Use of Human Memory Models: the Memory Extender Personal Filing System. *The International Journal of Man-Machine Studies* 25, 191-228

Lamming, M., and Flynn, M. 1994. "Forget-me-not:" Intimate Computing in Support of Human Memory. In Proceedings of FRIEND21, '94 International Symposium on Next Generation Human Interface, Meguro Gajoen, Japan.

Salton, G. ed. 1971. *The SMART Retrieval System -- Experiments in Automatic Document Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Starner, T., Mann, S., Rhodes, B., Healey, J., Russell, K., Levine, J., and Pentland, A. 1995. Wearable Computing and Augmented Reality, Technical Report, Media Lab Vision and Modeling Group RT-355, MIT

Wickens, C. 1992. *Engineering Psychology and Human Performance.* Scott Foresman Little Brown.

# Margin Notes:
# Building a Contextually Aware Associative Memory

Bradley J. Rhodes
MIT Media Lab
20 Ames Street
Cambridge, MA 02139 USA
+1 617 253 9601
rhodes@media.mit.edu

## ABSTRACT

Both the Human Computer Interaction and Information Retrieval fields have developed techniques to allow a searcher to find the information they seek quickly. However, these techniques are designed to augment one's direct-recall memory, where the searcher is actively trying to find information. Associative memory, in contrast, happens automatically and continuously, triggering memories that relate to the observed world. This paper presents design techniques and heuristics for building "remembrance agents," applications that watch a user's context and proactively suggest information that may be of use. General design issues are discussed and illuminated by a description of Margin Notes, an automatic just-in-time information system for the Web.

## Keywords

Contextual interfaces, software agents, remembrance agents, World Wide Web, browsers

## 1. INTRODUCTION

When using a tool or solving a problem, people will use a combination of their own knowledge and knowledge inherent in the environment. Through the use of affordances and other design techniques, a good design will endow the environment with as much information as possible to aid a user in solving their problem. However, no design can provide information for every situation, and no designer can include personalized information for every user. For example, a designer of a web page might add a hyperlink defining difficult terms or concepts in a project proposal, but they cannot add links letting a particular reader know that this proposal is similar to a project being implemented by the reader's own workgroup.

One solution to this problem is to create a "remembrance agent" (RA) that continuously watches a person's environment and provides information relevant to that person and their current situation. The provided information might be from a task-specific database, such as an address database, or from a more free-form information source such as personal email archives. For example, an early form of the remembrance agent software [9] automatically reminded a user of personal email and note files relevant to what was currently being typed into a word processor. Depending on what information is provided, an RA can act as a personal memory system, an organizational memory or a general just-in-time information system. [1]

Remembrance agents are similar to search engines, notification software, and personalized newspapers in that all provide personal information to a user, but there are also important differences. Examining these similarities and differences can inform the creation of design techniques for building RAs.

Like search engines, RAs provide information from a rich database given fuzzy and incomplete knowledge about what information should be presented. However, search engines and structured knowledge bases such as Yahoo! are inherently interactive: an information-seeker has some query in mind and directly interacts with the program to obtain the desired information. Remembrance agents, on the other hand, are proactive. The user need not have a query in mind, or even know that information relevant to her situation exists. An RA gives information without any prompting, so users need not know that useful information exists in their given context. However, receivers of this unrequested information may be less willing to take the time to process it, and are less willing to be distracted from their chosen task.

Notification software, like RAs, is proactive. Most of the time an alarm sits in the background and doesn't interrupt its user. When a new piece of email arrives, a stock price goes below a certain threshold, or another trigger event occurs, the alarm activates and informs the user. Both RAs and alarms also have the potential to be context-aware, using the user's own context to trigger information. However, RAs are designed to provide much richer and more varied kinds of information than alarms. Alarms usually provide only a single piece of information, namely whether the alarm's trigger event has occurred. Even if the algorithm to decide whether to trigger an alarm is very complex, the end result is by definition only one of a very limited set of states. On the other hand, a single RA might provide information from any of

[1] RAs are "software agents" in that they watch an environment and can act based on that environment without direct user intervention. They should not be confused with what are often called user interface agents or with distributed agent architectures.

thousands of documents, emails, or database entries with each "suggestion" leading to pages of information.

From a design standpoint, the most important difference between notification software and RAs is that a user is expected to know what an alarm means when she sees it. In some cases that knowledge is trivial. For example, it is obvious that you have new email when a letter appears in the iconic representation of your mailbox. In other cases that knowledge is the result of years of training, such as the knowledge required for a pilot to know what it means when a "oil pressure low" warning light flashes. Because RAs can show a variety of different pieces of information, the user cannot be expected to know what a particular suggestion contains. The display interface for an RA must be able to present that information.

Finally, RAs are similar to personalized newsfeeds that automatically send email when news that fits a personal profile is announced. Both systems are personalized and proactive, and both provide rich information from a large source pool. However, personalized newspapers aren't related to the receiver's current context. Instead, the personalization comes from manually or automatically generated user profile, while the trigger event to send a news article is usually far outside the receiver's personal environment. For example, a personal news system might watch the AP newswire and automatically compare articles with a subscriber's personal profile. When a close match is found, the news article is automatically sent to the subscriber's email. In some systems, the headline and first paragraph are also sent to the user's alphanumeric pager. However, the "sensors" in these systems are all looking at the newswire, not at the user. Personalization comes from a static user profile. In RAs the situation is reversed. The sensors of the system watch the user and the user's current context, and the personalization comes from these dynamic user-oriented sensors. Additional personalization occurs when drawing suggestions from a personal information database such as email archives.

The next section will introduce the Margin Notes system, a remembrance agent that on the fly annotates web pages with relevant information from a user's own email or other text files. The system has been in operation for two years as iterative user testing has been performed. Evaluations of the system are presented, and the system will then be used to illuminate design techniques for creating remembrance agent software.

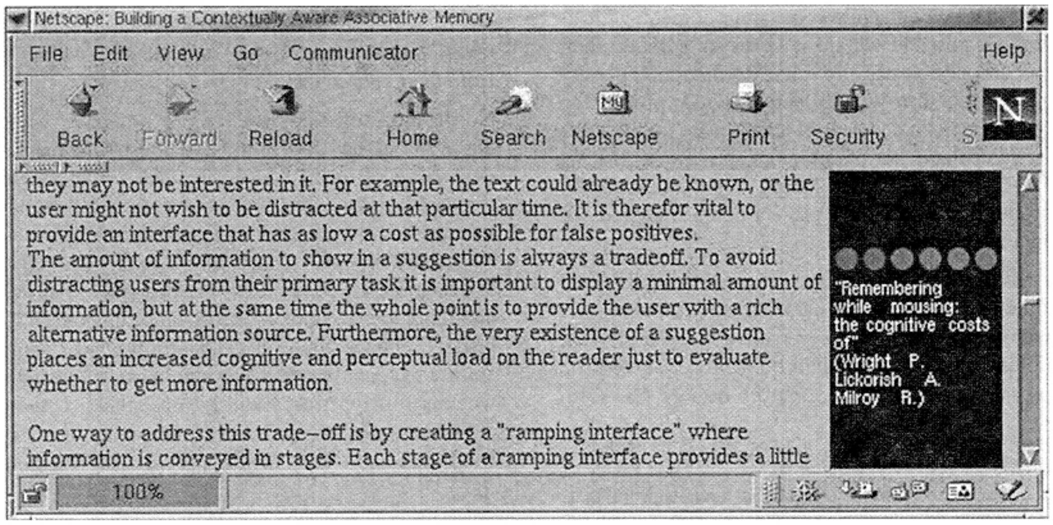
## 2. MARGIN NOTES

### 2.1 Overview

Margin Notes is a remembrance agent that automatically rewrites web pages as they are loaded, adding hyperlinks to personal files. As a web page is loaded, Margin Notes adds a black margin strip to the right of the document. It then compares each section of the document to pre-indexed email archives, notes files, and other text files, based on keyword co-occurrence. If one of these indexed files is found to be relevant to the current section of the web page, a small "suggestion box" is included in the margin next to the relevant section. The box contains a quick description of the suggested text, an bar representing the relevance of the suggestion, and a link to get more information. The suggestion box usually consists of a subject, date, author, and filename for the suggested text.

Placing the cursor over the suggestion box produces a list of the five keywords that were most important in deciding a suggestion's relevance to the annotated web page section. This list appears both in the message area at the bottom of the browser and as a pull-down window under the box itself.

Clicking on a suggestion box jumps to the full text of the email, note-file, or text being suggested. The suggested page also has feedback buttons used for evaluating the system.



they may not be interested in it. For example, the text could already be known, or the user might not wish to be distracted at that particular time. It is therefor vital to provide an interface that has as low a cost as possible for false positives.
The amount of information to show in a suggestion is always a tradeoff. To avoid distracting users from their primary task it is important to display a minimal amount of information, but at the same time the whole point is to provide the user with a rich alternative information source. Furthermore, the very existence of a suggestion places an increased cognitive and perceptual load on the reader just to evaluate whether to get more information.

One way to address this trade–off is by creating a "ramping interface" where information is conveyed in stages. Each stage of a ramping interface provides a little

●●●●●●
"Remembering while mousing: the cognitive costs of"
(Wright P. Lickorish A. Milroy R.)

Figure 1: A screenshot of the web version of this paper, annotated by Margin Notes. The database used for this example is a subset of the INSPEC database of paper abstracts. The suggested document, listed in the black margin to the right, is a paper from SigCHI Bulletin that discusses the display of information and cognitive load. The suggested document was previously unknown by the author, but is now being added to his thesis reading list.

## 2.2 Internals

Before running Margin Notes, potentially relevant documents are indexed to allow quick retrieval. Documents can be from a generally useful database such as a collection of newspaper or journal articles, or from personal sources such as email or notes. Pages can also be indexed directly from the web. Documents are re-indexed nightly to incorporate new changes and additions.

The system is a proxy server, and acts as an intermediary between the Web-browser and the Web. Previous versions of the system annotated on the fly, which required that the browser not show sections of the requested web page until the annotation was computed. This delay proved unacceptable to test users, especially for long documents. In the new system, a page is immediately passed on to the browser with the full text intact, but with the addition of a black margin to the right of the text. As suggestions are computed they appear in the black margin next to the section they annotate.

After sending the document to the browser, the text of the document is broken into sections based on header, paragraph, list, horizontal rule and center HTML tags. Small sections are ignored, as are JavaScript, comments and HTML tags. The remaining sections are passed to a relevance engine called Savant, which uses text-retrieval techniques to find documents relevant to the current section.

Savant looks at each document in the indexed database and computes a relevance score relative to the processed section of the web page. This score is based on co-occurrence of words using a term frequency / inverse-document frequency (TFiDF) method [11] and the Okapi weighting scheme [12]. Savant returns the document number for the five most relevant documents, along with their relevance score and a brief summary-line. Savant also compares title and header fields in the HTML to the equivalent fields in the returned document. For example, the header for the section of the web page being visited will be compared to the title of journal articles in the INSPEC database, and this relevance will be averaged into the overall score for the suggestion.

In previous versions of the system, users complained that while suggestions were often related to the section they annotated, no suggestion related to the big picture. To solve this problem, the first annotation on the page is now calculated using the entire web document as a query, with the first section is weighted more highly than the rest of the document. The remaining annotations are more focused and still use a single section as their query.

Of the documents suggested by Savant, the highest relevance document that has not already been suggested for this web page is selected. If the relevance score is above a threshold a suggestion-box is created and inserted into the margin of the web page at the top of the relevant section. The relevance threshold can be changed depending on the source database, and is adjustable for individual users.

When a suggestion link is followed, a script is run that retrieves the document and sends it to the web browser. The document might be reformatted based on its type; for example email headers are printed in a different font. For evaluation purposes, the script also logs the fact that a suggestion was followed, and inserts buttons at the top of the screen allowing the user to rate the suggestion.

## 3. DESIGN ISSUES FOR CONTEXTUALLY RELEVANT JUST-IN-TIME INFORMATION

The most important design constraint for remembrance agents is that reading suggestions must be a secondary task for a user. Unlike users of a search engine, users of remembrance agents are not actively seeking the information being suggested. They are therefor less tolerant of being distracted from their primary, chosen task, and are less willing to dig through suggestions to get useful information. Furthermore, even if a suggested text is highly relevant to the user's current context they may not be interested in it. For example, the text could already be known, or the user might not wish to be distracted at that particular time. It is therefor vital to provide an interface that has as low a cost as possible for false positives.

The amount of information to show in a suggestion is always a tradeoff. To avoid distracting users from their primary task it is important to display a minimal amount of information, but at the same time the whole point is to provide the user with a rich alternative information source. Furthermore, the very existence of a suggestion places an increased cognitive and perceptual load on the reader just to evaluate whether to get more information.

One way to address this trade-off is by creating a "ramping interface" where information is conveyed in stages. Each stage of a ramping interface provides a little more information, at the cost of the user spending a little more of their attention to read and understand it. The idea is to get useful information to a user quickly, while at the same time allowing them to bail out on an unwanted suggestion with as little distraction as possible.

In a ramping interface, the user should always be able to get more information by going to the next stage, and the effort required to get to that stage should be proportional to the amount of information provided in the current stage. It should only require a simple action, such as moving the mouse to a target, for a user to go to early stages. Going to a later stage might require the user to pick a topic from a menu, trading off simplicity for increased control of what information is displayed.

For example, in the Margin Notes system the first stage is what happens when the system analyzes a section of a web page and decides not to leave any suggestion at all. At this stage, the input is simply a passive sensor watching the user's actions. No user action or attention is required to show the agent what to do. The output at this stage is nothing: the agent simply passes the HTML to the browser and continues.

When Margin Notes determines that a suggestion is beyond a certain relevance threshold, it automatically jumps to the second stage and shows the suggestion box. In keeping with the philosophy that the effort in jumping to the next stage should be commensurate with the amount of work or attention required by the current stage, no user thought or action is required to display the suggestion box.

At this point the user may ignore the suggestion entirely, and the interaction has cost nothing more than some screen real estate and a very minor cognitive load. If he wishes to go to the next stage, he can quickly view a graphical bar of filled-in circles indicating the relevance value for that suggestion. In earlier versions a two-digit number was used to indicate relevance. While the same information was available, it was much more difficult to find and process quickly. The relevance bar, on the other hand, only requires perceptual processing to determine the relevance of a suggestion.

The next stage is reached by reading the suggestion description, which requires further attention on the part of the user. Information in the suggestion box is as terse as possible to allow rapid scanning for content. The box also contains many different kinds of information to try to contextualize the suggestion. For example, when email is displayed the box contains subject, author, date and archive filename in the hopes that at least one of these will give a good indication of the suggestion's content.

At the fourth stage of the Margin Notes interface the system displays the most relevant keywords. Going to this stage requires physical action by the user (a mouse-over of the link), and gives an incremental increase in the amount of information returned. While keyword information could have been included in the original suggestion (reducing the system to a four-stage ramping interface) it was decided that this information made the suggestion-box too cluttered. To jump to the final stage, the user clicks on the link and gets the fully suggested text. At this point the user is completely committed to seeing the text, and has been distracted from their primary task. However, even at this stage the text is displayed in a separate window, allowing the user to keep their initial context on the screen. Furthermore, the suggestion is still at least related to the user's original context. Hopefully if the user got to this point the suggested text was worth the attention spent.

It should also be noted that a user will automatically visit each stage of the ramping interface through normal interaction. In the case of the Web, the natural action to get more information is to click on a link. This requires first seeing the link in one's peripheral vision, then seeing it in the fovea, then moving the mouse over the link and clicking. At each stage of this set of actions the ramping interface gives a little more information, allowing the user to bail out if he chooses.

When designing a ramping interface it is helpful to consider at what stage a user is likely to receive the information they need. In the Margin Notes system, it is assumed that most of the time users will receive information from the full suggested text, but that occasionally they will be reminded by the suggestion box itself and never need to follow the link. On the other hand, remembrance agents designed for a car or other attention-demanding environment might be designed such that users normally needn't go past the first stage of a suggestion.

Another important design consideration is how to indicate to which part of a user's context a suggestion is relevant. Even if the user's full context is constrained to a single web page, it still needs to be apparent whether a suggestion is relevant to a single paragraph, a section, or the entire web page. This indication can be by fiat, e.g. by simply declaring that all suggestions are chosen based on relevance to the whole body, or by indicating the relevance somehow in the suggestion. In the Margin Notes system, suggestions are defined to be relevant to the section next to which they appear, within the context of the whole page.

Finally, it is important to be able to distinguish a suggestion from the user's other context. In some interfaces this is trivial, either because suggestions always appear out-of-band or because suggestions look nothing like anything that would normally be seen. Augmented reality systems, for example, fall into the latter category because the quality of computer screens is not high enough to be mistaken as a part of the real world. This is not the case for the Margin Notes system, where HTML is actually modified before being shown. In this case, suggestions need to be "branded" in such a way that they are clearly distinguished from the web page being viewed. This branding is accomplished by placing all Margin Notes within its own black border, producing a natural split between the web document's space and the Margin Notes space.

Equally important is what information to show, or indeed whether to show any information at all. In some applications, what to show is relatively trivial. For example, the Emacs Big-Brother Database automatically displays the contact information for the sender of any email received. Because the program is limited in scope, what to show is determined by a simple database lookup.

The task of finding relevant information is much harder when the information being suggested is richer than a simple database. In remembrance agents where both the context and suggested information is text, many techniques can be borrowed from the Information Retrieval community. However, selecting what information to show is more difficult than returning best matches to a search-engine query for several reasons.

The first complication is that remembrance agents need to not only return a best match, but must also determine whether to report a suggestion at all. This task is a combination of information filtering and information retrieval, and requires modification to many of the standard search-engine techniques. Because users are less willing to search through multiple suggestions for a good hit, it is also important that the search technique used be very precise, with a high likelihood of getting a good hit in the first few results returned.

Another potential complication is that remembrance agents are not basing their suggestions on actual user-generated queries, but only on passively detected user context. This throws all available information into a query, without the user being able to specify useful content first. Similarly, it can be difficult to detect context shifts, so suggestions may refer to something in which the user is no longer has an interest. On the other hand, the system shouldn't focus to narrowly on the exact context at hand or it will never see the forest for the trees. Margin Notes compromises by performing query expansion based on relevant keywords in the overall document, but still annotating individual sections of the web page.

Finally, the value of a remembrance agent is highly dependent on the source of suggestions. First, it is much easier to provide terse descriptions of a suggested text for personal information than non-personal, simply because the user has probably already seen the text before and knows the people and events involved. News and technical articles are less easy to contextualize, but have still been written for a general audience with a known general background. News and technical articles also have the advantage that their titles usually are written to convey the general idea of the full text. The hardest kind of information to convey is non-personal random information like web pages, where a title such as "Bob's Home Page" gives little information as to the page's actual content. Second, suggestions from high-quality, personally relevant corpora of information naturally tend to produce better suggestions. If the database is a poor match then at best no suggestions will be shown, at worst irrelevant and useless suggestions will be displayed. While this is equally true for search engines, a bad suggestion for a search engine is far less of an annoyance than for a remembrance agent. As was described by one beta-tester, when a search engine returns bad hits he just tries different search terms or another engine. When Margin Notes produces a bad suggestion he is annoyed that it cluttered his screen without merit.

## 4. EVALUATION

The usefulness of Margin Notes, and remembrance agents in general, depends greatly on how well their database meshes with user contexts and users themselves. Given a narrowly constrained task domain it is often possible to find a corpus containing only high-quality information. However, the ultimate goal of this research is to produce a system that will provide useful information in everyday situations, given data that is readily available. Task-performance analysis is still possible within these constraints, but it becomes necessary to show both that the tasks presented are typical of real-world tasks, and further that the database used is general enough that it will be useful for a wide range of tasks.

It is also difficult to evaluate Margin Notes based completely on the relevance of suggestions it provides, as might be done for an information-retrieval system. The problem is that relevance (or more accurately, "precision" and "recall") do not necessarily indicate usefulness of a suggestion. For example, a suggestion might be extremely relevant, but if the user has already seen the text, or if the suggested text is incorrect, or if the user just doesn't want to be distracted then the suggestion is not useful. As a real world example, Margin Notes will occasionally report a suggestion with an extremely high relevance. In some of these cases, the text is an exact copy of the web page being viewed, with no extra information at all. Certainly relevance is still a heuristic for usefulness, but usefulness should also be evaluated on it's own.

Margin Notes has been used by up to five people at a time for over two years, and a previous version was tested with long-term twelve users. Of the testers, three ran personal copies of Margin Notes that pulled suggestions from their own email archives, four ran personal copies that pulled suggestions from all their computer files (including email archives), and five ran a generic copy that pulled suggestions from the lab-wide email archive. The personal databases ranged from 762 unique files to almost 30,000 unique files. The generic database was the largest with over 115,000 emails ranging back to 1986. After the month-long test, the beta-testers were asked to fill out surveys giving their opinions on the system. Eight of the twelve original beta-testers returned surveys. Logging data tracked how many suggestions were displayed per page and which suggestions were followed. Results from this evaluation were mixed, and many of the problems that were uncovered have been addressed in the new version.

Of the 4576 suggestions shown during the period in question, only 6% of them were followed to the full-text. The most likely explanations, based on interviews of the users, are that users found most suggestions to be irrelevant or useless to their current situation, or never read the suggestion note to start with. Indeed, when asked to rank suggestion quality, speed of loading, not cluttering web pages with lots of suggestions, and making sure to show a relevant suggestion if one exists, all but one of the eight respondents said the most needed improvement for the current Margin Notes system was suggestion quality. When asked to rank the same four features in terms of importance for a Margin Notes system in general, all ranked suggestion quality in the top two (five as most important, three respondents ranking it second). This result has since lead to a complete rewrite of the Savant search engine with a new algorithm that uses more information from the current web page and indexed documents.

Users were also asked to rate on a five-point scale how often the suggestion box was a useful reminder in itself, such that they needn't look at the full-suggested text. With five being "very often," users gave an average answer of 3.17 (standard dev = 1.33).

However, in spite of the poor click-through on suggestions, overall user response was positive. When asked to rate on a five-point scale how likely they were to use the system after the experiment was over (five being very likely), the average result was 3.63 (standard dev = 0.74), with none of the eight respondents giving less than a three.

These evaluations have motivated further changes to the system, and new evaluations are being conducted to see the effects of these improvements.

## 5. RELATED WORK

Margin Notes is most related to other systems that automatically annotate documents with hyperlinks to more information. One such system is the Peace, Love, and Understanding Machine (PLUM) system [3]. PLUM will automatically add hyperlinks to disaster news stories to better give a reader a personalized understanding of the news. Another is VOIR [4], which combines elements of a document to be annotated with a user's interest profile to automatically create hyperlinks. A third is Watson [1], which automatically produces and submits an Alta-Vista query based on a user's current web page or Microsoft Word document. Results are clustered and displayed in a separate window.

Also similar are systems that automatically link to recommended documents based on a user profile. One such system is Letizia [7]. Letizia automatically creates a user profile by compiling keywords contained in previous web pages viewed by the user. It then acts as an "advanced scout," following links from the user's current and subsequent web pages and bringing up pages that match that profile. Letizia is in many ways the reverse of Margin Notes. Margin Notes generates "queries" based on a user's current web page, and applies this query to a relatively static database of potentially relevant documents. Letizia, on the other hand, generates queries based on a slowly changing personal profile built up from past browsing habits. This query is then applied to the set of documents linked off of the web page currently being visited. Another similar system is WebWatcher [5], which highlights existing hyperlinks that best match a user's stated interest. Like with Letizia, recommendations are only pulled from links off the page currently being viewed, based on a user profile. Recommended links are highlighted with a special icon.

Margin Notes also shares interface features with new methods for displaying hyperlinks. For example, the XLibris pen-based system [8] supports the ability to hand-annotate documents using margin links. Similarly, the Fluid Links system [13] allows web designers to create "glosses," which are a way of implementing a ramping interface.

Margin Notes is also similar to the Forget-Me-Not system [2][6], which is also designed to organize personal information without user intervention. However, Forget-Me-Not is designed primarily as a direct memory aid, allowing users to search a database of their own automatically recorded past actions. It falls into the same category as search engines and other interactive information retrieval systems, in that it is not designed to be proactive.

Finally, the Margin Notes work was based on previous work examining remembrance agents in word-processors [9] and on wearable computers [10].

## 6. FUTURE WORK

Based on the initial feedback, the next step is to improve the quality of suggestions produced by the Margin Notes system. Currently, short sections in web pages are more likely to produce hits than large sections, because there are fewer keywords that have to match. The next version will fix this problem. Finally, the number of beta-testers will be extended to get more data points for analysis, using both task-performance analysis and logging of long-time usage.

## 7. ACKNOWLEDGEMENTS

I'd like to thank my sponsors at British Telecom and Merrill Lynch, and Jesse Rabek, Jesse Koontz, and Jan Nelson, without whom this project would never have gotten off the ground. Finally, I'd like to thank Rich Devaul and Alan Wexelblat for their numerous suggestions and edits.

## 8. REFERENCES

[1] Budzik, J., and Hammond, K. Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science.* 1999.

[2] Brown, P., Bovey, J., and Chen, X. Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications, 4(5), October 1997.

[3] Elo, S. PLUM: Contextualizing News for Communities Through Augmentation. Thesis for Master of Science in Media Arts and Sciences, MIT, 1995.

[4] Golovchinsky, G., What the Query Told the Link: The Integration of Hypertext and Information Retrieval, in The Proceedings of HyperText 97, Southampton UK, 1997.

[5] Joachims, T., Freitag, D. and Mitchell, T. WebWatcher: A Tour Guide for the World Wide Web, in Proceedings of IJCAI97, August 1997.

[6] Lamming, M., Brown, P., Carter, K., Eldridge, M., Flynn, M., Louie, G., Robinson, P., and Sellen, A. The design of a human memory prosthesis. The Computer Journal, 37(3), 153-163. 1994.

[7] Lieberman, H. Autonomous Interface Agents, in Proceedings of CHI-97, Atlanta, Georgia, March 1997, ACM Press, 67-74.

[8] Price, M., Golovchinsky, G., Schilit, B., Linking by Inking: Trailblazing in a Paper-like Hypertext, in The Proceedings of HyperText 98, Pittsburgh, PA, June 98, pp. 30-39.

[9] Rhodes, B. and Starner, T. The Remembrance Agent: A continuously running automated information retrieval system, in The Proceedings of PAAM '96, London, UK, April 1996, pp. 487-495

[10] Rhodes, B. The Wearable Remembrance Agent: A system for augmented memory, Personal Technologies Journal Special Issue on Wearable Computing, Personal Technologies (1997), 1:218-224.

[11] Salton, G. Automatic Text Processing. The Transformation, Analysis and Retrieval of Information by Computer. Reading, MA, Addison-Wesley.

[12] Walker, S., Robertson, S.E., Boughanem, M., Jones, G.J.F., Sparck-Jones, K., Okapi at TREC-6, in The Sixth Text Retrieval Conference (TREC-6). Edited by D.K. Harman, Gaithersburg, MD: NIST, 1998.

[13] Zellweger, P., Chang, B., Mackinlay, J. Fluid Links for Informed and Incremental Link Transitions, in The Proceedings of HyperText 98, Pittsburgh, PA, June 98, pp. 50-57.

# Using Physical Context for Just-in-Time Information Retrieval

Bradley Rhodes

**Abstract**—Jimminy is a wearable personal note-taking and note-archival application that automatically displays notes that might be relevant to the wearer in his current environment. The system selects old notes to show on a head-up display based on the wearer's current location, people in the immediate area, and the subject-line and contents of any current notes being written. This paper describes an experiment that evaluates the usefulness of the wearer's physical context (location and people in the area) for automatically finding useful archived information. The results suggest that, while physical context can be used to discover useful archived notes, the subject and text of notes currently being entered are a much better indicator of usefulness in the personal note-taking domain.

**Index Terms**—Wearable computers, intelligent agents, information retrieval, context.

## 1 Introduction

A Just-in-time Information Retrieval (JITIR) agent is software that proactively retrieves and presents information based on a person's local context. JITIRs can be thought of as automatic "query-free" search engines. Rather than using a human supplied query, the JITIR uses elements of a person's environment (physical or computational) as an automatically executed query into a database of potentially useful information. Any information that passes a relevance threshold is automatically presented to the user, usually via an interface that minimizes distraction should the information not be relevant after all.

There are several desktop-based applications that meet the definition of a JITIR, ranging from domain-specific to general information applications. One domain-specific application is FixIt [1], a system that delivers repair-manual entries to a technician working with diagnostic software for copier repair. Another is Coach [2], which provides automatic help with LISP programming within the context of an integrated development environment. Lumiere [3], the basis for the Microsoft Office Assistant, is an example of a more general JITIR that provides help using a large number of Microsoft products based on Bayesian user models. More general still are Letizia [4], which suggests links to follow based on a user's Web-browsing activities, Watson [5], which automatically performs Web searches based on text being written or read in Microsoft Word, the Remembrance Agent [6], which suggests personal email and documents based on text being written or read in Emacs, and SUITOR [7], which displays stock market quotes, headlines, and other information based on text in a word processor or Web browser.

All the applications described above select information to display based on a user's computational environment: text being written or read or interactions with a particular application. Wearable and mobile computers provide the opportunity to perform the same kind of query-free retrieval of information using the wearer's physical context such as his location, people in the area, or date instead of only using the user's computational environment. However, most of the wearable and mobile applications of this kind are quite domain-specific and focus on providing information that is especially suitable for indexing by the physical features used in the system.

Location is an especially popular physical feature for automatic presentation of information on a wearable system. Examples include the Touring Machine [8] and Cyberguide [9], which automatically provide information about nearby places of interest in a city using GPS data. Similarly, the CIS Archeologist Field Assistant [10] uses GPS data to automatically retrieve field data collected about giraffes within a Kenyan game reserve. On a smaller scale, several wearable and hand-held systems use location to deliver information about exhibits at a conference or museum [11], [12], [13].

Systems using physical context other than location have also been developed. The DyPERS system [14] also presents information about museum exhibits, but, instead of location, uses machine vision to detect what painting a wearer of the system is currently viewing. The Augment-able Reality system uses two-dimensional bar codes attached to objects to bring up information whenever that object is in view [15]. Finally, systems are being developed to display biographical information about a person based on face-recognition machine-vision software [16].

The mobile and wearable systems described above all use a particular physical context to automatically retrieve information intimately associated with that context. It is not clear, however, how the use of physical context as an automatic query can generalize to broader domains. In particular, it is not clear whether physical context such as a person's location or people in the area is useful for automatically retrieving personal notes in an office setting or from a person's daily life.

The Jimminy system, also known as the Wearable Remembrance Agent, was designed to test the utility of physical context cues for automatically retrieving information in a personal note-taking system. Unlike the mobile systems described above, Jimminy displays information that is not explicitly chosen to fit well with the physical context being used for retrieval. The question being addressed is how well physical context can work when the association between information and physical context in which the information was obtained is only implicit, as in the case of personal notes in an office or home environment.

### 1.1 Jimminy (the Wearable Remembrance Agent)

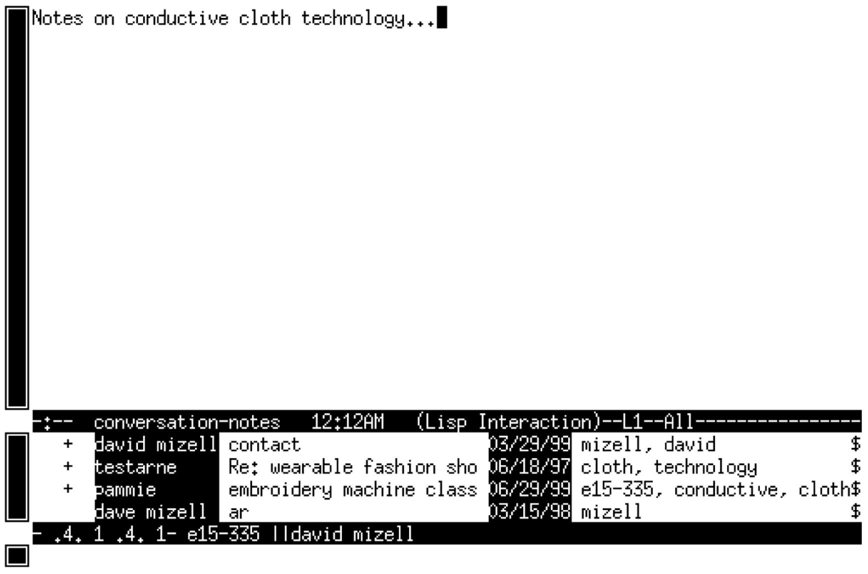
Jimminy is a wearable system that senses a wearer's location, people in the area, and text being entered on a wearable computer, and automatically displays information that might be relevant in that environment on a head-up display. The program has two concurrently running components, the note-taking environment and the automatic retrieval environment. Both components operate within the Emacs text-editor running on a Lizzy wearable computer [17], outfitted with a Private Eye head-up display, a Twiddler one-handed chording keyboard, a radio-frequency receiver, and an infrared-receiver. Jimminy is based on a desktop version called the Remembrance Agent [18], [6], which has the same basic functionality but does not use any physical sensors. An earlier version of Jimminy is described in [19].

Notes are entered using the one-handed keyboard and can be touch-typed at a rate between 35 and 50 words per minute. New notes are automatically tagged with a header indicating the wearer's current location, people in the area, and time and date. A subject line can be manually entered. Notes are displayed in the top 20 lines of the $320 \times 240$ monochrome head-up display, using 80 percent of the screen real-estate.

The automatic retrieval component of Jimminy continuously watches the environment (wearer's location, people in the immediate vicinity, subject of any notes being written, and the text of any notes being written) and automatically displays up to five one-line summaries of past notes that might be relevant to the

• *The author is with Ricoh Innovations, 2882 Sand Hill Rd., Menlo Park, CA 94025-7054. E-mail: rhodes@bradleyrhodes.com.*

*Manuscript received 1 July 2002; revised 1 Dec. 2002; accepted 1 Feb. 2003. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 118309.*



```
Notes on conductive cloth technology...

-:--  conversation-notes   12:12AM   (Lisp Interaction)--L1--All----------------
   +  david mizell contact                  03/29/99 mizell, david              $
   +  testarne     Re: wearable fashion sho 06/18/97 cloth, technology          $
   +  pammie       embroidery machine class 06/29/99 e15-335, conductive, cloth$
      dave mizell  ar                       03/15/98 mizell                     $
- .4. 1 .4. 1- e15-335 ||david mizell
```

Fig. 1. The Jimminy agent screenshot.

current situation, given the sensed context (see Fig. 1). When the environment is changing (for example, when the wearer moves to a new location or takes notes), the list of suggestions is updated every five seconds. The head-up display is positioned such that suggestions can be scanned easily but are not distracting. If desired, the full text of a suggested note can be retrieved with a single chord on the one-handed keyboard.

Notes are suggested based on similarity to the current environment. For example, upon entering a room, notes that were previously taken in that room are suggested. When a new person enters the room, notes that were previously taken with that person present will also be shown, with preference to notes where both the location and person match. As a note is typed, the text of the note is compared to the text of previous notes using a Term Frequency/inverse Document Frequency (TF/iDF) algorithm that retrieves notes on a similar topic [20]. The four different dimensions (location, people, subject, and text) are all used to compute relevance and the four similarity results are linearly combined to form an overall ranking of potentially relevant documents. Earlier versions of Jimminy also used date, time, and day-of-week as context information, though more recent versions do not. Dimensions that have recently changed (for example, "location" when a new room has just been entered) receive extra weight when computing similarity.

Jimminy detects the wearer's location and people in the area using radio location beacons and infrared active name badges. The active name badges use the Locust IR beacon [21], which has a directional range of a few feet. Location beacons are Locusts that have been modified to transmit radio instead of infrared and are omnidirectional with a radius of about 10-15 feet. At the time of the experiment, location beacons were placed in several rooms around the MIT Media Lab, with one or two per room. Whenever the wearer of the system comes into range of a beacon, Jimminy automatically updates its record of the user's location and who is in the area. Sensors are integrated using a Java-based distributed agent architecture called Hive [22], [23], which handles the low-level beacon protocol and translation from beacon IDs into location and person names.

Jimminy was in use at the MIT Media lab for several years, but beacons and active badges were not always available and were not the primary focus of the research. Location beacons were not distributed beyond a few rooms within the MIT Media Lab and never outside the lab. Active name badges were not used except for demonstrations. To handle cases when location and person beacons were not available, physical information could be entered manually using the one-handed chording keyboard. The keyboard was also used to enter notes and to give each note a title or subject line. Time and date-stamps were automatically added to a note as it was created. Due to the lack of beacon coverage, manual entry was by far the most common way Jimminy received knowledge about the wearer's physical context.

## 2 Evaluation

Jimminy is designed for the recording and suggesting of personal notes, and collections of such notes take time to build and will be greatly affected by the environment in which they are taken. To understand how the system might be used over long periods of time in real-world environments, Jimminy was worn and used by the primary researcher in his daily life over the course of several years. This was a part of the MIT Media Lab living experiment in wearable computing.

From 1996 to 2000, Jimminy was used daily by the author and more than 850 notes were written and annotated on topics ranging from classes and conversations at conferences to descriptions of dance steps. Of these, 664 were tagged with information about the physical context in which the note was taken. The remaining notes were plain-text files without additional tags.

The quantitative evaluation of Jimminy was designed to test the value of suggestions and, in particular, to test whether information about location and people in the area improved suggestions.

### 2.1 Method

Six different sets of paired notes were generated using the Jimminy automatic-retrieval system: Location, Person, Subject, Note-text, All-features, and Control group. Each set was generated by processing notes that have already been written through the Jimminy automatic-retrieval system as if that note was just being entered. The first note of each pair was designated the "query" note, the note suggested by Jimminy was the result note. Before generating the suggestion pairs, the weights for Jimminy were adjusted so only the desired pieces of context would be used in generating a suggestion. For example, pairs in the Location set were generated only by looking only at similarity in the location field of a note's header. Similarly, the Note-text set was generated by taking every note and pairing it with the top suggestion

TABLE 1
Scores for Each Set

| | Number scores tallied | | | | | |
|---|---|---|---|---|---|---|
| Set | 1 | 2 | 3 | 4 | 5 | Percentage scores 4 or 5 |
| All Features | 16 | 4 | 2 | 11 | 17 | 56% |
| Note Text | 16 | 2 | 7 | 11 | 14 | 50% |
| Subject | 32 | 5 | 1 | 5 | 7 | 24% |
| Location | 37 | 6 | 1 | 2 | 4 | 12% |
| Person | 40 | 3 | 3 | 2 | 2 | 8% |
| Control | 45 | 3 | 2 | 0 | 0 | 0% |

generated by Jimminy when all physical context and the subject was ignored. The All-features set was generated by pairing each note with whatever note would normally be suggested by Jimminy, using all the physical context as well as the subject and text of the note. The Control group consisted of random pairings. For each set, notes that were blank in the given field were removed. For example, the Person set had only notes that were tagged with at least one person.

A test suite was then created by taking 50 pairs at random from each of the six sets, for a total of 300 pairs. Pairs were then presented in random order and without labels to the researcher who originally took the notes. The researcher evaluated each pair for usefulness based on the following question: "If one of these notes were being written right now, how likely would it be for the other note to be useful?" Notes were rated one through five, with one being "definitely useless," four being "probably useful," and five being "definitely useful."

The primary advantage to the methodology described above is that evaluations are performed using both "query" environments and a notes database that actually occurred during long-term use. Furthermore, the fact that notes were taken over the course of several years makes it possible to evaluate long-term effects such as returning to the same classroom for a new class during a different school-year. However, the need for long-term usage also makes it difficult to run tests with multiple subjects. The results described here are only drawn from a single subject: the researcher himself while he was a 26-30-year-old graduate student. Results would likely differ for notes taken by subjects with other backgrounds, such as traveling salesmen or law enforcement authorities.

It should also be noted that the real usage of Jimminy differs from the test described above in a few important ways. First, Jimminy shows not one but five suggestions at any one time, but this experiment only rated the topmost suggestion that would be displayed. Second, if a suggestion's similarity rating is below a certain threshold, then Jimminy does not display it at all. This experiment scored all top suggestions, even those that would not have been displayed due to poor confidence in the suggestion. Also, cases where no suggestion could be found were automatically given the lowest score. Third, suggestions that were computed using the body of a note (the All-features set and the Note-text set) were computed using the entire text of the note. In actual use, these suggestions would only appear after the user had typed in the entire note; suggestions that would be displayed when the note was only halfway entered may be different. Finally, this experiment was conducted over the entire notes collection, which means notes might be suggested that were actually taken after the "query" note had been written. In an actual system, the first notes written about a new topic will necessarily produce no related notes. Only after other notes are taken on the same topic will useful suggestions arise.

TABLE 2
Breakdown of Location and Person Features

| | Location | Person |
|---|---|---|
| Notes with feature not blank | 558 | 466 |
| Notes with unique feature (singleton) | 89 | 167 |
| Notes with most common | 74 | 33 |
| Notes with 2nd most common | 65 | 27 |
| Notes with 3rd most common | 55 | 11 |
| % Notes with unique feature | 16% | 36% |
| % Notes with most common | 13% | 7% |
| % Notes with one of two most common | 25% | 13% |
| % Notes with one of three most common | 35% | 15% |

## 2.2 Results and Discussion

Table 1 lists scores tallied for each of the six groups and the percentage of pairs that were rated either "probably useful" or "definitely useful." The difference between the five experimental groups and the control group are all statistically significant ($p = 0.05$). The difference between the All-features set and the Note-text set was not significant.

Several conclusions can be drawn from this experiment. First, it is clear that the standard algorithm for Jimminy (the All-features set) produces useful suggestions, at least for this particular user and this particular set of notes. This is a promising result and supports the user's subjective opinion that the system produced useful suggestions during actual use.

Second, it is clear that the location in which a note was taken and people who were present at the time are not nearly as useful as the contents of a note or the subject line for determining what information might be useful to display. While a hit rate of around 10 percent may be good enough for some applications, it is still disappointing.

Third, the use of location, person, subject, and note text did not significantly improve the relevance of suggestions beyond that obtained using only the note text. This lack of improvement was not entirely due to the physical criteria having no effect on suggestions: 40 percent of the queries gave different results in the All-features condition than they did in the Note-text condition.

One likely cause for the poor performance is that the location and people in the area are poor distinguishing features for this particular set of notes. To be a good feature for distinguishing between notes, a feature should neither occur only once (such that no suggestions are produced) nor should it occur so frequently that the system is left with too many notes to choose from after filtering based on the feature. As a graduate student, the researcher spent very little time outside of his office or the two classrooms within the Media Lab. He also tended to meet a large number of people at conferences, demos, or talks that were never seen again.

Table 2 shows the breakdown of how frequently locations and people appeared in the corpus of notes. Locations were especially clumped, with over a third of the notes being taken in one of three places. These locations were, in order of frequency, the common area just outside the note-taker's office, his office itself, and the main classroom for the Media Lab. This is similar to problems reported with using location as a cue for query-based retrieval in the PEPYS system [24], where many unrelated activities occurred when the system user was alone in his office [25]. The people feature had the reverse problem: Over a third of the notes taken were tagged with a person that never appeared again in any other notes. This is likely due to frequent sponsor demos and guest lecturers that occur at the Media Lab. In both these situations, a person was often met once and never seen again.

While the distribution of locations and people within the notes database clearly contributes to the poor showing for these features, there is another explanation: The location where a note was taken and people present are simply a poor indicator of the topic of notes taken for this experiment. The assumption is that notes on the same topic as the wearer's current conversation, lecture, meeting, or idea will be useful. The problem is that "topic" is hard to define and even harder for a machine to sense and represent. The features that Jimminy actually uses (location, person, subject, and note text) are all used as proxies for the topic of the wearer's current thought. It stands to reason that the best representation of this topic is the text of the notes he is taking at the time, especially since this feature is the richest representation of the four. It also stands to reason that the subject line of a note will be a reasonably good proxy for topic since the subject line is composed by the note-taker for that purpose. The person and location fields do not have the richness of the note text nor are they chosen specifically by the note-taker to be good representations of topic. This does not mean the physical context cannot still be a good proxy for topic and, indeed, it is a good proxy in specific domains such as museum tours. However, it does mean that the task domain and especially the corpus of notes from which suggestions are drawn must be chosen carefully to insure physical context will be useful for automatic retrieval of useful information.

## 3 Conclusion

Jimminy is an example where physical context, namely, location and people in the vicinity, is not especially useful for automatically retrieving information from a personal notes archive. However, given that the experiment described was conducted with only one subject, it is dangerous to generalize these results too far. As was stated in the introduction, the specific circumstances in which notes are taken will have a major impact on their structure and, thus, the usefulness of any one feature. What can be generalized are the lessons for identifying where physical context might be useful in such a system.

First, features should neither be so sparse that there are a large number of unique occurrences of a feature nor should they be clumped. Sparseness is a problem because it is unlikely that a match will be found at all for a sparsely distributed feature. Clumping is a problem because the feature will not distinguish between potentially useful suggestions. The concept that rare but nonunique features are best for retrieving information is well understood in the information-retrieval field and is used in most search-engine algorithms today [20].

Second, physical features need to correlate well with the topic in which a person is currently interested and the topic of a note that might be suggested. For example, the person feature in a salesman's notes may correlate very strongly with the topic because sales activities tend to center on people. Location may correlate well with topic in the notes of a tourist, museum patron, or police officer.

Finally, designing just-in-time information retrieval agents requires a good understanding of both the contexts in which they will be used (that is, the "query" the system will use) and the corpus from which suggested information will be drawn. By understanding the task domain, the system can be based on features that correlate well with the underlying topic of a note or user's environment.

## Acknowledgments

The author would like to thank Professor Thad Starner for his feedback and advice throughout this research.

## References

[1] P.E. Hart and J. Graham, "Query-Free Information Retrieval," *Proc. Conf. Cooperative Information Systems,* pp. 36-46, 1994.

[2] T. Selker, "Coach: A Teaching Agent that Learns," *Comm. ACM,* vol. 37, no. 7, pp. 92-99, 1994.

[3] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," 1998.

[4] H. Lieberman, "Letizia: An Agent that Assists Web Browsing," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 924-929, 1995.

[5] J. Budzik and K. Hammond, "Watson: Anticipating and Contextualizing Information Needs," *Proc. 62nd Ann. Meeting Am. Soc. for Information Science,* pp. 727-740, 1999.

[6] B. Rhodes and P. Maes, "Just-in-Time Information Retrieval Agents," *IBM Systems J.,* vol. 39, no. 3, pp. 685-704, 2000.

[7] P. Maglio, R. Barret, C. Campbell, and T. Selker, "Suitor: An Attentive Information System," *Intelligent User Interfaces,* ACM Press, pp. 169-176, 2000.

[8] S. Feiner, B. MacIntyre, and T. Höllerer, "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment," *Proc. First Int'l Symp. Wearable Computers,* pp. 74-81, Oct. 1997.

[9] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A Mobile Context-Aware Tour Guide," *ACM Wireless Networks,* vol. 3, pp. 421-433, 1997.

[10] J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers," *Proc. Second Int'l Symp. Wearable Computers,* pp. 92-99, Oct. 1998.

[11] F. Sparacino, "The Museum Wearable: Real-Time Sensor-Driven Understanding of Visitors' Interests for Personalized Visually-Augmented Museum Experiences," *Proc. Museums and the Web (MW 2002),* Apr. 2002.

[12] R. Oppermann, M. Specht, and I. Jaceniak, "Hippie: A Nomadic Information System," *Lecture Notes in Computer Science,* vol. 1707, pp. 330-333, 1999.

[13] Y. Sumi, T. Etani, S. Fels, N. Simonet, K. Kobayashi, and K. Mase, "C-Map: Building a Context-Aware Mobile Assistant for Exhibition Tours," *Community Computing and Support Systems,* 1998.

[14] B. Schiele, T. Jebara, and N. Oliver, "Sensory Augmented Computing: Wearing the Museum's Guide," *IEEE Micro,* 2001.

[15] J. Rekimoto, Y. Ayatsuka, and K. Hayashi, "Augment-Able Reality: Situated Communications through Physical and Digital Spaces," *Digest of Papers: Second Int'l Symp. Wearable Computers,* pp. 68-75, Oct. 1998.

[16] A. Pentland and T. Choudhury, "Face Recognition for Smart Environments," *Computer,* vol. 33, no. 2, pp. 50-55, Feb. 2000.

[17] T. Starner, "Wearable Computing and Context Awareness," PhD dissertation, MIT Media Laboratory, Cambridge, Mass., May 1999.

[18] B. Rhodes and T. Starner, "Remembrance Agent: A Continuously Running Automated Information Retrieval System," *Proc. Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM),* Apr. 1996.

[19] B. Rhodes, "The Wearable Remembrance Agent: A System for Augmented Memory," *Personal Technologies J.,* special issue on wearable computing, vol. 1, no. 4, pp. 218-224, 1997.

[20] D. Harman, *Information Retrieval: Data Structures and Algorithms,* pp. 363-392. Prentice Hall, 1992.

[21] T. Starner, D. Kirsch, and S. Assefa, "The Locust Swarm: An Environmentally-Powered, Networkless Location and Messaging System," Technical Report 431, MIT Media Lab, Perceptual Computing Group, Apr. 1997.

[22] N. Minar, M. Gray, O. Roup, and P.M. Raffi Krikorian, "Hive: Distributed Agents for Networking Things," ASA/MA '99, 1999, http://hive.media.-mit.edu/.

[23] B. Rhodes, N. Minar, and J. Weaver, "Wearable Computing Meets Ubiquitous Computing: Reaping the Best of Both Worlds," *Proc. IEEE Int'l Symp. Wearable Computers,* pp. 141-149, 1999.

[24] W. Newman, M. Eldridge, and M. Lamming, "PEPYS: Generating Autobiographies by Automatic Tracking," *Proc. European Conf. Computer Supported Cooperative Work,* pp. 175-188, 1991.

[25] M. Eldridge, M. Lamming, and M. Flynn, "Does a Video Diary Help Recall?" *Proc. Conf. People and Computers VII,* A. Monk, D. Diaper, and M.D. Harrison, eds., pp. 257-269, 1992.